SERVER SELECTION METHOD

BACKGROUND OF THE INVENTION

The present invention pertains to communication systems and more particularly to server availability for pooled server architectures.

In modern communication systems, clients typically receive periodic advertisement from servers of a common pool indicating their availability and current traffic loading. The advertisement includes the status of each of a number of pooled servers. The client then typically performs a probabilistic selection from the pool based on the server's capacity. For example, if three servers had a capacity of 10, 40 and 50 call units, they would be selected approximately 10%, 40% and 50% of the time respectively.

If one of the servers from the pool is unavailable due to either network or server difficulties, the selection percentages remaining in effect until the network or server fault can be detected. If the frequency of a server's, or any other resource's, selection is significantly higher than the frequency with which the fault is detected, the faulty server or resource may be selected for use many times during the time period until the fault is detected.

During the faulty server time period, all attempted selections of the faulty server will lead to an increase in call setup time. Further, continued selection failures could result in loss of the call by the communication system. Such loss of the call may affect revenue by the communication system operator and sub-standard call services, as measured by a public utility commission, being supplied to a number of communication system users.

Further, unnecessary retries of server selection during a fault condition cause an increase in network and server central processing unit (CPU) utilization without calls being

successfully connected. Such situation could exacerbate a transient server fault and network congestion.

Accordingly, it would be highly desirable to have a server lottery selection method which provides a low cost method for a client to dynamically adjust the probability of server selection during faulty conditions.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a communication system in accordance with the present invention.

FIG. 2 is a flow chart of a server lottery selection method in accordance with the present invention.

FIG. 3 is a state diagram of a server lottery selection method in accordance with the present invention.

PREFERRED EMBODIMENT OF THE INVENTION

Referring to FIG. 1, a distributed computing network 12 is shown. A client 10 communicates with one or more servers 30, 40, and 50 via a communication network 20. The communication network 20 provides unreliable message delivery service (i.e., under heavy load or upon receipt of corrupted messages, the network can and will discard messages). The communication network 20 does not provide any explicit indication of the resource availability of the attached network elements (clients such as 10 and servers 30, 40 and 50).

The exchange of resource availability data must be performed by the network elements (client 10 and servers 30, 40 and 50) themselves. However in the case of a server fault or network fault, the ability for the clients and servers to communicate is compromised. Such faults lead to inaccurate data feeding into the lottery selection method shown below in FIG. 2.

A server fault is defined as any condition that renders the server incapable of fulfilling its role in the network, e.g., a software failure results in the server being offline while the server reboots; a hardware failure results in the server being offline until the failed hardware can be replaced.

A network fault is defined as any condition that renders the network incapable of providing message delivery service for the clients and servers, e.g., hardware faults, software faults, and network congestion cause messages to be discarded, which effectively disconnects the communication paths between the clients and servers.

Referring to FIG. 2, a flow chart diagram of the lottery selection method with the dynamic recovery improvement is shown. Client 10 is ready to accept inbound events at block 60. Upon receipt of an event at block 62, the lottery selection method 61 determines and handles one of three event types.

Upon receipt of a server resource advertisement 67, control is transferred from block 62 to block 68. At block 68, the new server resource data is written to the resource database 66.

Server resource advertisements indicate the type and magnitude of resources that a server is able to provide to the client. For example, a telephone network server advertises to clients that it has the capacity to support 500 telephone calls with a current load of 150 active telephone calls. Clients 10, 11, etc. use this data to effectively distribute the load of future telephone calls across the available telephone network servers.

Upon receipt of a server allocation request 63, control is transferred from block 62 to block 64. Block 64 selects a server (30, 40 or 50) using the resource database 66 and the lottery selection method 61. Block 64 transfers control to block 65 to generate a random number between zero (0) and one (1). Then the selection is made as explained below. For

example, client 10 has received the following resource advertisements 67:

Server 30: total capacity = 500; active load = 150.
Server 40: total capacity = 600; active load = 200.
Server 50: total capacity = 700; active load = 250.

The available network capacity equals the total server capacity minus the total active load:

Total Server Capacity = 500 + 600 + 700 = 1800
Total Active Load = 150 + 200 + 250 = 600
Available Network Capacity = 1800 - 600 = 1200

Using the lottery selection method 61, client 10 makes a probabilistic selection of a server:

Server 30 has an available capacity of 500 - 150 = 350. Therefore server 30 should be selected with probability = 350 / 1200 = 29.2%.

Server 40 has an available capacity of 600 - 200 = 400. Therefore server 40 should be selected with probability = 400 / 1200 = 33.3%.

Server 50 has an available capacity of 700 - 250 = 450. Therefore server 50 should be selected with probability = 450 / 1200 = 37.5%.

A computer (not shown) of the client 10 generates a random number between 0 and 1. The client 10 selects the server as follows:

If random number <= 0.292, then select server 30 (29.2%)

If 0.292 < random number <= 0.625, then select server 40 (33.3%)

Otherwise, select server 50 (37.5%)

Lottery scheduling is a randomized resource allocation mechanism. Resource rights are represented by lottery tickets. Each allocation is determined by holding a lottery; the resource is granted to the client with the winning ticket. This effectively allocates resources to competing clients in proportion to the number of tickets that they hold.

Lottery tickets encapsulate resource rights that are abstract, relative, and uniform. They are abstract because they quantify resource rights independently of machine details. Lottery tickets are relative, since the fraction of a resource that they represent varies dynamically in proportion to the contention for that resource. Thus, a client will obtain more of a lightly contended resource than one that is highly contended; in the worst case, it will receive a share proportional to its share of tickets in the system. Finally, tickets are uniform because rights for heterogeneous resources can be homogeneously represented as tickets. These properties of lottery tickets are similar to those of money in computational economies.

Scheduling by lottery is probabilistically fair. The expected allocation of resources to clients is proportional to the number of tickets that they hold. Since the scheduling algorithm is randomized, the actual allocated proportions are not guaranteed to match the expected proportions exactly. However, the disparity between them decreases as the number of allocations increases.

The number of lotteries won by a client has a binomial distribution. The probability P that a client holding t tickets will win a given lottery with a total

of T tickets is simply P = t / T. After N identical lotteries, the expected number of wins W is E[W] = NP, with variance V[W] = NP(1-P). The coefficient of variation for the observed proportion of wins is SQRT(NP(1-P)). Thus, a client's throughput is proportional to its ticket allocation, with accuracy that improves with SQRT(N). A single physical ticket may represent any number of logical tickets. This is similar to monetary notes, which may be issued in different denominations.

The number of lotteries required for a client's first win has a geometric distribution. The expected number of lotteries N that a client must wait before its first win is E[N] = 1 / P, with variance V[N] = (1 - P) / P^2. Thus, a client's average response time is inversely proportional to its ticket allocation. The properties of both binomial and geometric distributions are well-understood.

With a scheduling quantum of 10 milliseconds (100 lotteries per second), reasonable fairness can be achieved over sub-second time intervals. As computation speeds continue to increase, shorter time quanta can be used to further improve accuracy while maintaining a fixed proportion of scheduler overhead.

Since any client with a non-zero number of tickets will eventually win a lottery, the conventional problem of starvation does not exist. The lottery mechanism also operates fairly when the number of clients or tickets varies dynamically. For each allocation, every client is given a fair chance of winning proportional to its share of the total number of tickets. Since any changes to relative ticket allocations are immediately reflected in the next allocation decision, lottery scheduling is extremely responsive.

Upon receipt of a failed server allocation event 69, block 62 transfers control to block 70. The server allocation

failure event 69 results from either the server explicitly denying service or the server failing to respond to the allocation request within a predefined period of time.

The faulted server's available capacity metric in the resource database 66 is decreased 70 by a scaling factor. Using the example above for servers 30, 40 and 50 and illustrating a failure of server 40, the available capacity metric for server 40 is reduced from 400 to 200. This decrease in the capacity metric also decreases the overall network available capacity from 1200 to 1000. The probability of server selection is updated as follows.

Following Failure #1:

Server 30: probability = 350 / 1000 = 35%.

Server 40: probability = 200 / 1000 = 20%.

Server 50: probability = 450 / 1000 = 45%.

After handling any of these events 67, 63, or 69, control is transferred from blocks 68, 64, or 70, respectively to block 60 to wait for the next event. The client 10 is then ready to accept the next event 60.

Referring to FIG. 3, a state diagram is shown. The state diagram 99 depicts the transitions between normal operation 80 and failure operation 90.

Normal operation is defined as the condition where periodic server resource advertisements are being received and server allocation requests are being serviced successfully. Failure operation is defined as the condition where a server has failed to service one or more allocation requests, which results in a reduction in the probability of selecting that server.

The client 10 begins in normal operation 80. Upon receipt of a resource data advertisement 67, the client 10 updates the database 66 and continues normal operation 80 via path 82. While in normal operation 80, the distributed computing network has all server resources available. Also the clients

10, 11, etc. are balancing the offered load across the pool of servers.

Upon receipt of a server allocation failure event 69, the client 10 enters failure operation 90 via path 84. Each allocation failure event 69 causes the server's available capacity metric to be reduced by a scaling factor, which reduces the probability of the faulted server being selected during the next lottery selection.

Consecutive server allocation failure events 69 via path 92 result in an exponential decay in the probability of selecting the affected server.

Continuing the example above, a second allocation request failure for server 40 would further reduce its available capacity from 200 to 100, which also decreases the network available capacity from 1000 to 900. The probability of server selection is updated as follows.

Following Failure #2:
Server 30: probability = 350 / 900 = 38.9%.
Server 40: probability = 100 / 900 = 11.1%.
Server 50: probability = 450 / 900 = 50%.

Additional allocation request failures continue to drive the exponential decay in the probability of selecting server 40 as follows.

Following Failure #3:
Server 30: probability = 350 / 850 = 41.2%.
Server 40: probability = 50 / 850 = 5.9%.
Server 50: probability = 450 / 850 = 52.9%.

Following Failure #4:
Server 30: probability = 350 / 825 = 42.4%.
Server 40: probability = 25 / 825 = 3.0%.
Server 50: probability = 450 / 825 = 54.5%.

Following Failure #5:

Server 30: probability = 350 / 813 = 43.1%.

Server 40: probability = 13 / 813 = 1.6%.

Server 50: probability = 450 / 813 = 55.4%.

With a typical scaling factor of 0.5, ten consecutive failures would reduce the probability of selecting the affected server by a factor of approximately $0.5 \wedge 10 \approx .001$. This is due to the exponential nature of the decrease. Each allocation request failure reduces the server's available capacity metric by a factor of 0.5, and consecutive reductions are compounded, which results in a rapid decrease.

In an alternate embodiment, the scaling factor could be driven from a static table with a particular number of entries to accelerate or slow the effects of a fault. For example the table may include entries such as the following (e.g., 0.75, 0.50, 0.25, etc). Instead of multiplying by a fixed scaling factor, the scaling operation included in the data base 66, for example, and reference the static table to determine how much to reduce the available capacity metric for each detected fault.

During a server fault or network fault, the disclosed method enables the client to detect server allocation request failures (detection) and dynamically adjust the selection criteria (adjustment), which results in resource allocation requests being diverted away from the faulted server and toward the remaining available servers. In the example above, the probability of selecting server 40 dropped from 33.3% to 1.6% in five failure detections; under moderate client load, this adjustment would occur in less than one second.

Upon receipt of a resource data advertisement 67 from a faulted server, the new available resource capacity metric supercedes all previous reductions of that server's metric, which results in an immediate convergence to normal operation 80 via path 94 (recovery). For example, when a server

recovers from a hardware or software fault, it will advertise its available capacity to the clients, and the clients will immediately begin using that data (i.e., the decay in probability is exponential during failure operation 90, but return to normal operation 80 is instantaneous).

Although the preferred embodiment of the invention has been illustrated, and that form described in detail, it will be readily apparent to those skilled in the art that various modifications may be made therein without departing from the spirit of the present invention or from the scope of the appended claims.